Swap elements

Anna wants to know what is the minimum number of element swapping required to make non-empty zero-indexed array A of N integers sorted ascending.

For example, given the following array:

A[0] = 1
A[1] = 2
A[2] = 6
A[3] = 5
A[4] = 5
A[5] = 8
A[6] = 9

it is possible to do only one swap to make array sorted:

- swap A[2]=6 with A[4]=5

For the following array:

A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 20
A[4] = 5
A[5] = 6
A[6] = 7

It is required to make at least 3 swapping. One possible way it can be done is:

- swap A[3]=20 with A[4]=5
- swap A[4]=20 with A[5]=6
- swap A[5]=20 with A[6]=7

Write a function:

```
object Solution { def solution(a: Array[Int]): Int }
```

which, given non-empty zero-indexed array A of N integers, returns the minimal number of swapping required to make array sorted ascending.

Assume that:

- N is an integer within the range [1..100,000]
- each element of array A is an integer within the range [-100,000,000..100,000,000]

Complexity:

- expected worst-case time complexity is O(N*log(N))
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input array can be modified.